Announcement

- My $2 bill continues to search for a new owner!
- No lectures this coming week, focus on the project.

Design and Implementation of V (Term Project)

Shahram Ghandeharizadeh
Computer Science Department
University of Southern California

Outline

- Objectives
- Specification of V
- A design

Objectives

- Design and implement a cache manager for a hierarchical storage system that stores and retrieves objects with variable sizes.
- An architecture:

V Functionalities
- Cache Replacement
- BDB-Disk
- BDB-Mem

Homework 2

Objectives

- Design and implement a cache manager for a hierarchical storage system that stores and retrieves objects with variable sizes.
- An architecture:

V Functionalities
- Initialize(int MemSizeInGBytes, int MemSizeInBytes)
- Shutdown()
- Create(const Vdt& dzname)
- GetNumberOfDataZones()
- GetZoneNames(Vdt *dzNameArr, int *NumberOfDataZones)
- Insert(const Vdt& dzname, const Vdt& Key, const Vdt& Value)
- Delete (const Vdt& dzname, const Vdt& Key)
- Get (const Vdt& dzname, const Vdt& Key, Vdt *Value)
- DeleteAllKeys (const Vdt& dzname)
- DeleteInAllDataZones(const Vdt& *KeyArray, const int size)
V Functionalities (Highlights)
- Designs MUST use the Vdt data structure.
- Enables us to implement test routines that will exercise different implementations.
- A default data-zone name, “_V_Universal”.
- Design and implement a technique to cache data into BDB-Mem (Mem).

Sample Test
- Initialize V with 1 GB,
- Insert 32K objects with Keys 0, 1, 2, ..., 31999 into V:
  - Values are strings of arbitrary length.
  - Shut the database down,
- Initialize V with 1 GB,
- Get all objects with Keys ranging from 0, 1, 2, ..., 31999,
  - Verify all Key/Value pairs exist in V,
  - Shut the database down,
- Initialize V with 1 GB,
  - Instantiate a Zipfian distribution to generate values ranging from 0 to 31999,
  - Generate ten thousand requests and measure the elapsed time,
  - Shut the database down
  - Report the average object retrieval time.
- The implementation with the fastest retrieval time wins an award!

BDB-Disk
- One instance of Berkeley DB.  What to store in it?

V Functionalities
Cache Replacement
BDB-Disk BDB-Mem

BDB-Disk, Disk for short
- One instance of Berkeley DB.  What to store in it?
  - A Catalog database to maintain the name of data-zones and its associated meta-data (if any).
  - A database for every data-zone as it is created.
  - A database for “_V_Universal” to store Key/Value pairs.
    - Hash or B-tree database?
    - When do create this database?

Cache Replacement
- One suggestion: GreedyDual

```c
GreedyDual(Clip x)
if (clip x is cache resident)
(Q) = \begin{cases}
\text{Clip (x)} & \text{if free space} < \text{Size}(x) \\
\text{Clip (x)} & \text{else}
\end{cases}
\text{while (free space < Size}(x)\text{)}
\text{L} = \text{min}(\text{Q})
\text{Exist } j \text{ from local cache,}
\text{Retrieve and store clip x,}
\text{Q} = (L + \text{Size}(x))
```

GreedyDual
- What are the key implementation details?

```c
GreedyDual(Clip x)
if (clip x is cache resident)
(Q) = \begin{cases}
\text{Clip (x)} & \text{if free space} < \text{Size}(x) \\
\text{Clip (x)} & \text{else}
\end{cases}
\text{while (free space < Size}(x)\text{)}
\text{L} = \text{min}(\text{Q})
\text{Exist } j \text{ from local cache,}
\text{Retrieve and store clip x,}
\text{Q} = (L + \text{Size}(x))
```
GreedyDual

- What are the key implementation details?
  - Find the object with minimum Qx value efficiently to select victims.
  - Update Q value of a referenced object that observes a cache hit efficiently.

A Design

- Maintain 3 databases in BDB-Mem:
  - Cache_DB:
    1. Key = DZname:oid
    2. Value = Value stored in the persistent DB
    3. Hash index on Key
  - Q_DB:
    1. Key = Q:DZname:oid
    2. Value = Size
    3. B-tree index on Key
  - Rev_DB:
    1. Key = DZname:oid
    2. Value = Q
    3. Hash index on Key
Victim Selection

1. Let \( F \) = size of the incoming object.
2. Let \( S_Z = 0 \).
3. Open a cursor on \( Q_{DB} \) and retrieve the smallest \( Q:DZname:oid \).
4. \( S_Z += \) Value from the cursor.
5. Delete \( DZname:oid \) from \( Cache_DB \).
6. Delete \( DZname:oid \) from \( Rev_DB \).
7. Delete current entry pointed to by the cursor.
8. If \( (S_Z >= F) \)
   - Close cursor and return.
9. Otherwise go to the next object in the cursor and repeat from Step 4.

Insert(NULL, Key, Value)

- Insert the Key/Value pair in "V:Universal" database in BDB-Disk.
- Insert \( DZname:Key/Value \) into \( Cache_DB \).
- \( Q = L + 1/\text{Size(Value)} \)
- Insert \( Q:DZname:Key/1 \) into \( Q_{DB} \).
- Insert \( DZName:Key/Q \) into \( Rev_DB \).

Is it a good idea to insert every incoming Key/Value into DRAM? If not then how do you decide what to keep in DRAM?

Thread Safe Code

- Protect your data structures using critical sections.

A Design

- When do you create these databases?
  1. \( Cache_DB \)
  2. \( Q_{DB} \)
  3. \( Rev_B \)

V Functionalities

Cache Replacement

- BDB-Disk
- BDB-Mem

Common (anticipated) Errors

- A design document that uses "Update".
  - BDB does not support "Update"; it supports inserts and deletes.
- Retrieves values without specifying where they come from.
  - Retrieves minimum Q value without specifying a database for it.
- Presents a cache manager without describing organization of data in the disk.
- Focuses on issues that are not central to the project, e.g., focuses on how to improve the performance of inserts and deletes, forgetting about gets.
  - Once you have mastered gets then focus on inserts and deletes.
- Start implementation with an incomplete design document.
  - This makes a simple, fun project into a monster from a nightmare.
An Improvement

- What is a limitation of having 3 databases in main memory?
- How do you reduce the number of databases from 3 to 2?
- Should you partition each database into a smaller fragments?

Advanced Topics

- What is the consequence of using 1/Size as the value of Q?

Cache_DB:
1. Key = DZname:oid
2. Value = Value stored in the persistent DB
3. Hash index on Key

Q_DB:
1. Key = Q:DZname:oid
2. Value = Size
3. B-tree index on Key

Rev_DB:
1. Key = DZname:oid
2. Value = Q
3. Hash index on Key